Tomography

Generated by Doxygen 1.8.7

Mon Aug 13 2018 11:36:10

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 plot Namespace Reference

Variables

- tuple data_file = open(sys.argv[1], ’r’)
- tuple av_distances = np.array([0,0,0])
- list non_physical = [ ]
- list x_values = [ ]
- tuple lineM = data_file.readline()
- list M = lineM[1]
- tuple lineN = data_file.readline()
- list N = lineN[1]
- tuple lineS = data_file.readline()
- list S = lineS[1]
- tuple current = line.split(’,’)
- tuple lineT = data_file.readline()
- list total_time = lineT[1]
- tuple ax2 = ax1.twinx()
- tuple atext

3.1.1 Variable Documentation

3.1.1.1 tuple plot.atext

Initial value:

```
= AnchoredText("Number of purity parameters: " + str(M) + "\n"
                     +"Density matrices: "+ str(N) + "\n"
                     +"Samples per measurement: " + str(S) + "\n"
                     +"Total running time:  " + str(total_time),
                     loc=2)
```

3.1.1.2 tuple plot.av_distances = np.array([0,0,0])

3.1.1.3 tuple plot.ax2 = ax1.twinx()

3.1.1.4 tuple plot.current = line.split(’,’)

3.1.1.5 tuple plot.data_file = open(sys.argv[1], 'r')

3.1.1.6 tuple plot.lineM = data_file.readline()

3.1.1.7 tuple plot.lineN = data_file.readline()

3.1.1.8 tuple plot.lineS = data_file.readline()

3.1.1.9 tuple plot.lineT = data_file.readline()

3.1.1.10 list plot.M = lineM[1]

3.1.1.11 list plot.N = lineN[1]

3.1.1.12 list plot.non_physical = [ ]

3.1.1.13 list plot.S = lineS[1]

3.1.1.14 list plot.total_time = lineT[1]

3.1.1.15 list plot.x_values = [ ]

Chapter 4

File Documentation

4.1 enm-test.cpp File Reference

```
#include "enm-test.h"
```

Include dependency graph for enm-test.cpp:

Functions

- int main ()

4.1.1 Function Documentation

4.1.1.1 int main ( )

4.2 enm-test.h File Reference

```
#include <iostream>
#include <complex>
#include <cstdlib>
#include <ctime>
#include <random>
#include <fstream>
#include <iomanip>
#include <chrono>
#include "Eigen/Dense"
#include "simulation.h"
#include "estimation.h"
#include "stats.h"
#include "proj.h"
#include "progress.h"
```

Include dependency graph for enm-test.h: This graph shows which files directly or indirectly include this file:

Macros

- #define DEBUG_PRINT_DISTANCE_AVERAGES
- #define SHOW_PROGRESS

Typedefs

- typedef Eigen::Matrix
  < std::complex< double >
  , Eigen::Dynamic,
  Eigen::Dynamic > MatrixXc

4.2.1 Macro Definition Documentation

4.2.1.1 #define DEBUG_PRINT_DISTANCE_AVERAGES

4.2.1.2 #define SHOW_PROGRESS

4.2.2 Typedef Documentation

4.2.2.1 typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> **MatrixXc**

4.3 estimation.cpp File Reference

```
#include "estimation.h"
```

Include dependency graph for estimation.cpp:

Functions

- MatrixXc linear_estimate_XYZ (double X_data[], double Y_data[], double Z_data[], int S)
- double d (const std::vector< double > &x, std::vector< double > &grad, void ∗f_data)
- MatrixXc enm_estimate_XYZ (double X_data[], double Y_data[], double Z_data[], int S)

4.3.1 Function Documentation

4.3.1.1 double d (const std::vector< double > & *x*, std::vector< double > & *grad*, void ∗ *f_data*)

4.3.1.2 **MatrixXc** enm_estimate_XYZ (double *X_data[ ]*, double *Y_data[ ]*, double *Z_data[ ]*, int *S*)

4.3.1.3 **MatrixXc** linear_estimate_XYZ (double *X_data[ ]*, double *Y_data[ ]*, double *Z_data[ ]*, int *S*)

4.4 estimation.h File Reference

```
#include <iostream>
#include "stats.h"
#include "Eigen/Dense"
#include <vector>
#include "nlopt.hpp"
```

Include dependency graph for estimation.h: This graph shows which files directly or indirectly include this file:

Macros

- #define DEBUG_PRINT_ENM_OUTPUT

Typedefs

- typedef Eigen::Matrix
  < std::complex< double >
  , Eigen::Dynamic,
  Eigen::Dynamic > MatrixXc

Functions

- MatrixXc linear_estimate_XYZ (double X_data[ ], double Y_data[ ], double Z_data[ ], int S)
- MatrixXc enm_estimate_XYZ (double X_data[ ], double Y_data[ ], double Z_data[ ], int S)

4.4.1 Macro Definition Documentation

4.4.1.1 #define DEBUG_PRINT_ENM_OUTPUT

4.4.2 Typedef Documentation

4.4.2.1 typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> **MatrixXc**

4.4.3 Function Documentation

4.4.3.1 **MatrixXc** enm_estimate_XYZ (double *X_data[ ],* double *Y_data[ ],* double *Z_data[ ],* int *S*)

4.4.3.2 **MatrixXc** linear_estimate_XYZ (double *X_data[ ],* double *Y_data[ ],* double *Z_data[ ],* int *S*)

4.5 plot.py File Reference

Namespaces

- plot

Variables

- tuple plot.data_file = open(sys.argv[1], 'r')
- tuple plot.av_distances = np.array([0,0,0])
- list plot.non_physical = [ ]
- list plot.x_values = [ ]
- tuple plot.lineM = data_file.readline()
- list plot.M = lineM[1]
- tuple plot.lineN = data_file.readline()
- list plot.N = lineN[1]
- tuple plot.lineS = data_file.readline()
- list plot.S = lineS[1]
- tuple plot.current = line.split(',')
- tuple plot.lineT = data_file.readline()
- list plot.total_time = lineT[1]
- tuple plot.ax2 = ax1.twinx()
- tuple plot.atext

4.6 progress.cpp File Reference

```
#include "progress.h"
```

Include dependency graph for progress.cpp:

Functions

- int show_progress (std::chrono::time_point< std::chrono::steady_clock > start, double p)

4.6.1 Function Documentation

4.6.1.1 int show_progress (std::chrono::time_point< std::chrono::steady_clock > *start,* double *p*)

4.7 progress.h File Reference

```
#include <iostream>
#include <string>
#include <sstream>
#include <chrono>
```

Include dependency graph for progress.h: This graph shows which files directly or indirectly include this file:

Functions

- int show_progress (std::chrono::time_point< std::chrono::steady_clock > start, double p)

4.7.1 Function Documentation

4.7.1.1 int show_progress (std::chrono::time_point< std::chrono::steady_clock > *start,* double *p*)

4.8 proj.cpp File Reference

```
#include "proj.h"
```

Include dependency graph for proj.cpp:

Functions

- int make_projector (MatrixXc A, MatrixXc proj_A[], double outcomes_A[])

4.8.1 Function Documentation

4.8.1.1 int make_projector (MatrixXc *A,* MatrixXc *proj_A[],* double *outcomes_A[]*)

4.9 proj.h File Reference

```
#include <iostream>
#include "Eigen/Dense"
```

Include dependency graph for proj.h: This graph shows which files directly or indirectly include this file:

Typedefs

- typedef Eigen::Matrix
  < std::complex< double >
  , Eigen::Dynamic,
  Eigen::Dynamic > MatrixXc

Functions

- int make_projector (MatrixXc A, MatrixXc proj_A[ ], double outcomes_A[ ])

4.9.1 Typedef Documentation

4.9.1.1 typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> **MatrixXc**

4.9.2 Function Documentation

4.9.2.1 int make_projector (**MatrixXc** *A*, **MatrixXc** *proj_A[ ]*, double *outcomes_A[ ]*)

4.10 simulation.cpp File Reference

```
#include "simulation.h"
```

Include dependency graph for simulation.cpp:

Functions

- MatrixXc random_unitary (std::mt19937 &generator)
- MatrixXc random_density (double x, std::mt19937 &generator)
- int simulate (MatrixXc dens, const MatrixXc proj[ ], const double meas[ ], int S, double sim_dat[ ], std::mt19937 &generator)

4.10.1 Function Documentation

4.10.1.1 **MatrixXc** random_density (double *x*, std::mt19937 & *generator*)

4.10.1.2 **MatrixXc** random_unitary (std::mt19937 & *generator*)

Function: Generate random unitary

The method is to parametrise the unitary group and then select the right distribution for the parameters. See '2009 Ozols - How to generate a random unitary matrix', page 5, for more details.

4.10.1.3 int simulate (**MatrixXc** *dens*, const **MatrixXc** *proj[ ]*, const double *meas[ ]*, int *S*, double *sim_dat[ ]*, std::mt19937 & *generator*)

4.11 simulation.h File Reference

```
#include <iostream>
```

```
#include <iomanip>
#include <cstdlib>
#include <ctime>
#include <chrono>
#include <random>
#include "Eigen/Dense"
#include <cmath>
```

Include dependency graph for simulation.h: This graph shows which files directly or indirectly include this file:

Macros

- #define DEBUG_PRINT_RANDOM_DENSITY
- #define DEBUG_PRINT_MEASUREMENTS
- #define _USE_MATH_DEFINES

Typedefs

- typedef Eigen::Matrix
  < std::complex< double >
  , Eigen::Dynamic,
  Eigen::Dynamic > MatrixXc

Functions

- MatrixXc random_unitary (std::mt19937 &generator)
- MatrixXc random_density (double x, std::mt19937 &generator)
- int simulate (MatrixXc dens, const MatrixXc proj[ ], const double meas[ ], int S, double sim_dat[ ], std::mt19937 &generator)

4.11.1 Macro Definition Documentation

4.11.1.1 #define _USE_MATH_DEFINES

4.11.1.2 #define DEBUG_PRINT_MEASUREMENTS

4.11.1.3 #define DEBUG_PRINT_RANDOM_DENSITY

4.11.2 Typedef Documentation

4.11.2.1 typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> **MatrixXc**

4.11.3 Function Documentation

4.11.3.1 **MatrixXc** random_density (double *x*, std::mt19937 & *generator*)

4.11.3.2 **MatrixXc** random_unitary (std::mt19937 & *generator*)

Function: Generate random unitary

The method is to parametrise the unitary group and then select the right distribution for the parameters. See '2009 Ozols - How to generate a random unitary matrix', page 5, for more details.

4.11.3.3 int simulate (**MatrixXc** *dens,* const **MatrixXc** *proj[ ],* const double *meas[ ],* int *S,* double *sim_dat[ ],* std::mt19937 & *generator*)

4.12 stats.cpp File Reference

```
#include "stats.h"
```

Include dependency graph for stats.cpp:

Functions

- double distance_op (MatrixXc A, MatrixXc B)
- double distance_trace (MatrixXc A, MatrixXc B)
- double distance_fid (const MatrixXc A, const MatrixXc B)
- double distance_fid_2 (const MatrixXc A, const MatrixXc B)
- double mean (double array[ ], int N)

4.12.1 Function Documentation

4.12.1.1 double distance_fid (const **MatrixXc** *A,* const **MatrixXc** *B*)

4.12.1.2 double distance_fid_2 (const **MatrixXc** *A,* const **MatrixXc** *B*)

4.12.1.3 double distance_op (**MatrixXc** *A,* **MatrixXc** *B*)

4.12.1.4 double distance_trace (**MatrixXc** *A,* **MatrixXc** *B*)

4.12.1.5 double mean (double *array[ ],* int *N*)

4.13 stats.h File Reference

```
#include "iostream"
#include "Eigen/Dense"
#include "Eigen/SVD"
```

Include dependency graph for stats.h: This graph shows which files directly or indirectly include this file:

Macros

- #define DEBUG_PRINT_DISTANCES

Typedefs

- typedef Eigen::Matrix
  < std::complex< double >
  , Eigen::Dynamic,
  Eigen::Dynamic > MatrixXc

Functions

- double distance_op (MatrixXc A, MatrixXc B)
- double distance_trace (MatrixXc A, MatrixXc B)
- double distance_fid (const MatrixXc A, const MatrixXc B)
- double distance_fid_2 (const MatrixXc A, const MatrixXc B)
- double mean (double array[ ], int N)

4.13.1 Macro Definition Documentation

4.13.1.1 #define DEBUG_PRINT_DISTANCES

4.13.2 Typedef Documentation

4.13.2.1 typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> MatrixXc

4.13.3 Function Documentation

4.13.3.1 double distance_fid (const MatrixXc *A,* const MatrixXc *B*)

4.13.3.2 double distance_fid_2 (const MatrixXc *A,* const MatrixXc *B*)

4.13.3.3 double distance_op (MatrixXc *A,* MatrixXc *B*)

4.13.3.4 double distance_trace (MatrixXc *A,* MatrixXc *B*)

4.13.3.5 double mean (double *array[ ],* int *N*)

Index